High-Performance Component Technology

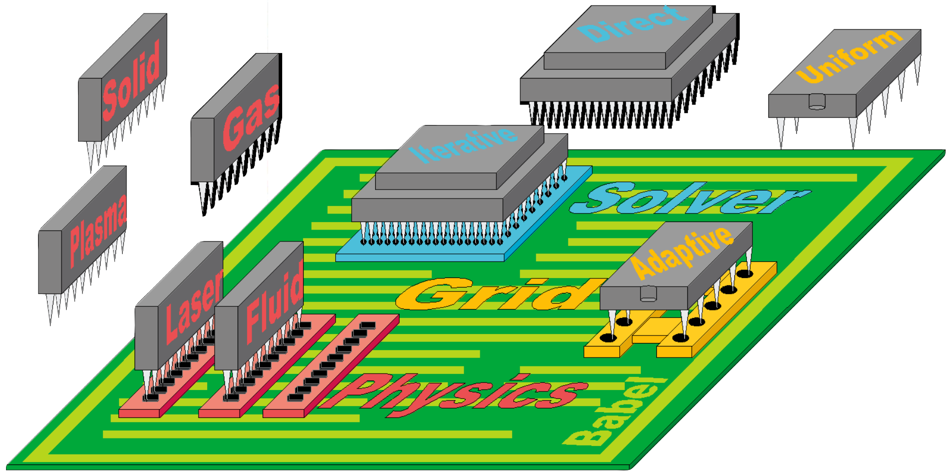


Figure 1: Component applications are built from component building blocks; components can be thought of as software "integrated circuits."

Mission

The High-Performance Components project develops software component technology for high-performance parallel scientific computing to address problems of complexity, re-use, and interoperability for laboratory simulation software. Our research focuses on the unique requirements of scientific computing on parallel machines, such as fast in-process connections among components, language interoperability for scientific languages, and data distribution support for massively parallel components.

The Need for Component Technology

Numerical simulations play a vital role in the DOE's science mission as a basic research tool for understanding fundamental physical processes. As simulations become increasingly sophisticated and complex, no single person—or even single laboratory—can develop scientific software in isolation. Instead, physicists, chemists, mathematicians, and computer scientists concentrate on developing software in their domain of expertise. Computational scientists create simulations by combining these individual software pieces.

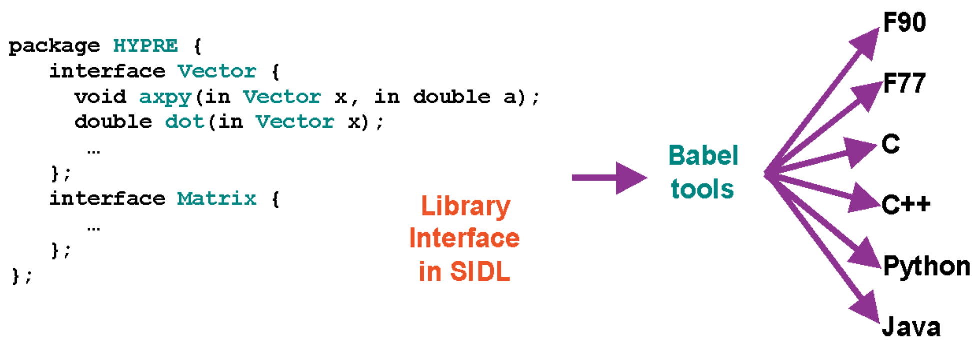
Unfortunately, it is often difficult to share sophisticated software packages among applications due to differences in implementation languages, programming style, or calling interfaces. In the industrial sector, this problem is solved through the use of component technology. Unfortunately, industry component solutions are inappropriate for parallel scientific computing because they do not support the concept of a "parallel component" that is required for high-performance scientific computing.

Scope of Project

We are investigating and developing component technology in three primary areas. First, we have developed a software tool called Babel that enables language interoperability among a variety of scientific programming languages, including Fortran, C, C++, Java, and Python. Second, we have developed a web-based component repository called Alexandria for deploying software components. Finally, we are investigating parallel data redistribution issues for communication among distributed components running on differing numbers of parallel processors.

Our Babel tool addresses language interoperability issues for high-performance parallel scientific software. Its purpose is to enable the creation, description, and distribution of language-independent software libraries. Babel uses Interface Definition Language (IDL) techniques. An IDL describes the calling interface (but not the implementation) of a particular software library. We have designed a Scientific Interface Definition Language (SIDL) that addresses the unique needs of parallel scientific computing. SIDL supports complex numbers and dynamic multidimensional arrays as well as parallel communication directives that are required for parallel distributed components. As shown in Figure 2, Babel uses this SIDL interface description to generate glue code that allows a software library implemented in one supported language to be called seamlessly from any other supported language. An important capability of Babel is that languages may be mixed together freely; libraries may be written in any supported language and called by any other supported language. For example, an application written in Python could call a solver library written in C, a physics package written in Fortran or C++, and a visualization package written in Java.

Babel currently supports Fortran 90, Fortran 77, C, C++, Python, and Java. Improving Fortran 90 support is our top priority this year. We are also researching extensions to SIDL that would add semantic descriptions for the behavior of scientific components beginning with basic precondition and post-condition assertions.



```
package HYPRE {
   interface Vector {
     void axpy(in Vector x, in double a);
     double dot(in Vector x);
     …
   };
   interface Matrix {
     …
   };
};
```

Figure 2: Babel provides language interoperability using SIDL interface descriptions.

We have also designed and implemented a prototype web-based repository called Alexandria to encourage the distribution and re-use of scientific software components and libraries. Alexandria provides a convenient web-based delivery system, and thus lowers the barrier to adopting component technology. We work with the DOE Common Component Architecture forum to establish common schema for accessing Alexandria from component tools developed by collaborators at other DOE laboratories and academia.

Finally, we are investigating the issues associated with parallel data redistribution among components running on different parallel machines and on different numbers of processors. Communication between two applications running on differing numbers of processors requires data redistribution. For example, a simulation code running on thousands of processors may need to communicate mesh data to a visualization server with a relatively small number of processors. Although the data redistribution problem has been addressed in the past for simple data types such as multidimensional arrays, we are investigating general techniques for the sophisticated data structures typically found in complex scientific applications, such as meshes and sparse matrices.

Common Component Architecture Forum

We are working closely with members of the Common Component Architecture (CCA) forum (see http://www.cca-forum.org). The CCA is a working group of physicists, mathematicians, and computer scientists developing component technology standards that address the high-performance computing needs of the DOE. CCA members include participants from the DOE (ANL, LANL, LBNL, LLNL, ORNL, and SNL) and academia (Indiana University and the University of Utah). The CCA is developing a reference implementation of a component technology infrastructure for high-performance computing. Our Babel tool provides the underlying language interoperability technology for the CCA infrastructure.

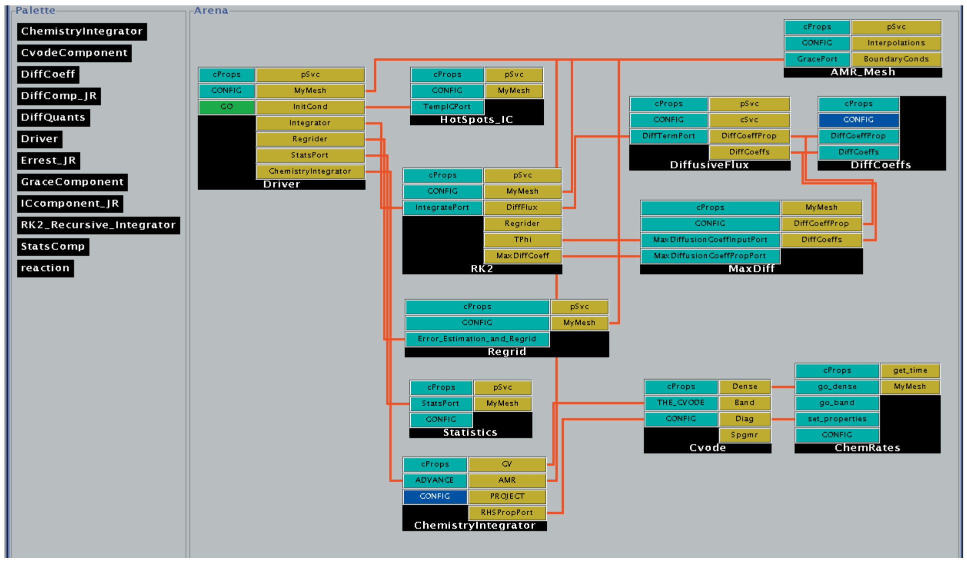


Figure 3: CCA component connections in Ccaffeine

Technology Demonstrations

We are collaborating with laboratory research groups to demonstrate component technology in scientific libraries and applications. These collaborations help us understand the issues involved in using advanced software technologies for scientific simulations, and they demonstrate the applicability of component approaches.

In particular, we are working with the hypre Scalable Linear Solvers team to integrate Babel language interoperability technology into their solver library. Our technology will enable the hypre library, developed using object-oriented techniques in C, to be called from scientific applications written in Fortran 77, Fortran 90, C, C++, Java, and Python.

The CCA has adopted SIDL as its official language for specification, and there are two implementations of the CCA specification using Babel language interoperability technology. We developed a prototype CCA framework called Decaf, and this work has been adopted into SNL's Ccaffeine, a CCA framework with a graphical user interface for composing simulations from a pallet of high-performance components.

The High-Performance Components project is funded by the SciDAC program in the DOE Office of Advanced Scientific Computing Research.

Contact Information:
Additional information about High-Performance Component Technology at LLNL is available from our web site http://www.llnl.gov/CASC/components. You may also contact the team at components@llnl.gov or Tom Epperly at tepperly@llnl.gov, 925-424-3159.

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.